

Edge Coloring with Minimum Reload/Changeover Costs

Didem Gözüpek¹ and Mordechai Shalom^{2,3}

¹ Department of Computer Engineering, Gebze Technical University, Kocaeli, Turkey. didem.gozupek@gtu.edu.tr *

² TelHai Academic College, Upper Galilee, 12210, Israel. cmshalom@telhai.ac.il

³ Department of Industrial Engineering, Boğaziçi University, Istanbul, Turkey. **

Abstract. In an edge-colored graph, a traversal cost occurs at a vertex along a path when consecutive edges with different colors are traversed. The value of the traversal cost depends only on the colors of the traversed edges. This concept leads to two global cost measures, namely the *reload cost* and the *changeover cost*, that have been studied in the literature and have various applications in telecommunications, transportation networks, and energy distribution networks. Previous work focused on problems with an edge-colored graph being part of the input. In this paper, we formulate and focus on two pairs of problems that aim to find an edge coloring of a graph so as to minimize the reload and changeover costs. The first pair of problems aims to find a proper edge coloring so that the reload/changeover cost of a set of paths is minimized. The second pair of problems aim to find a proper edge coloring and a spanning tree so that the reload/changeover cost is minimized. We present several hardness results as well as polynomial-time solvable special cases.

Keywords: Changeover cost, reload cost, approximation algorithms, edge coloring, network design, network optimization.

1 Introduction

1.1 Background

The cost incurred while traversing a vertex via two consecutive edges of different colors is called *traversal cost* and it depends only on the colors of the traversed edges. This cost leads to two different global cost measures that appeared in the literature under the names of *reload cost* and *changeover cost*.

* This work is supported by the Scientific and Technological Research Council of Turkey (TUBITAK) under grant no. 113E567.

** Supported in part by the TUBITAK 2221 Programme.

The reload cost concept is defined in [1] and it received attention only recently, although it has numerous applications. For instance, a color may represent the mode of transportation in an intermodal cargo transportation network. The traversal cost corresponds to the cost of transferring cargo from one carrier to another. Another application is in energy distribution networks, where the energy transfer from one carrier to another one, such as the conversion of natural gas from liquid to gas state, results in loss of energy. In telecommunications, traversal costs arise in numerous settings. For instance, routing in a heterogeneous network requires switching among different technologies such as cables, fibers, and satellite links. This switching cost can be modeled by traversal costs. Even within the same technology, switching between different providers, for instance switching between different commercial satellite providers in satellite networks, leads to a switching cost. All applications hitherto mentioned can be modeled using traversal costs where an edge-colored graph is given as input, and this is the focus of the works in the literature, e.g. [1–8]. However, problems of finding a proper edge coloring so as to minimize the reload (or changeover) cost have important applications as well. For instance, recently, cognitive radio networks (CRN) have gained increasing attention in the communication networks research community. Unlike other wireless technologies, CRNs are envisioned to operate in a wide range of frequencies. Therefore, switching from one frequency band to another frequency band in a CRN has a significant cost in terms of delay and power consumption [9]. An optimal allocation of frequencies to the wireless links in CRNs so that the switching cost is minimized, corresponds to a proper edge coloring minimizing the traversal cost. In this work we focus on this type of problems.

The reload cost refers to the sum of the traversal costs of a set of paths in a graph, whereas the changeover cost does not depend on the amount of commodity (or number of paths) traversing a vertex. The works [1, 2] consider the problem of finding a spanning tree having minimum diameter with respect to reload cost. The paper [3] considers the *minimum reload cost cycle cover* problem, which is to find a set of vertex disjoint cycles spanning all vertices, with minimum reload cost. In [4], the authors study the problem of finding a path, trail, or walk of minimum reload cost between two given vertices. They consider (a)symmetric reload costs and reload costs with(out) triangle inequality. The paper [6] studies the problem of finding a spanning tree that minimizes the sum of reload costs over the paths between *all* pairs of vertices. The paper [5] presents various path, tour, and flow problems related to reload costs. One of the

problems studied in that work is the *minimum reload cost path-tree*, which is to find a spanning tree that minimizes the reload cost from a given root vertex to all other vertices. The paper [7] studies a closely related, yet different problem, called *minimum changeover cost arborescence* in which the goal is to find a spanning tree that minimizes the changeover cost from a given root vertex to all other vertices. The work in [8] considers the same problem and derives several inapproximability results, as well as polynomial-time algorithms for some special cases. In [10] these special cases are extended to show that the problem is polynomial-time solvable in bounded treewidth graphs.

1.2 Our Contribution and Research Directions

In this paper, we consider a different problem and focus on proper edge coloring of a given graph such that the reload/changeover cost is minimized. To the best of our knowledge, this paper is the first study of this family of problems. Specifically, we formulate two pairs of problems. In the first pair of problems, given a set of paths comprising a graph, the goal is to find a proper edge coloring of the graph so that the reload (resp. changeover) cost is minimized. In the second pair of problems, given a graph and a root vertex, the goal is to find proper edge coloring and a spanning tree of the graph so that the reload (resp. changeover) cost from the root vertex to all other vertices is minimized. We present several hardness results as well as polynomial-time solvable special cases of these problems.

Specifically, we show that the first pair of problems are hard to approximate in general, within any polynomial-time computable function of the input length, and this result is valid when the traversal costs are unbounded. When the traversal costs are bounded, we prove that the problems remain NP-Hard even when the underlying network is a star. However, we prove that this pair of problems are polynomial-time solvable in trees when the degrees of the vertices are bounded.

We then prove that the second pair of problems are APX-Hard under bounded traversal costs in directed graphs. On the positive side, for these problems, we present a polynomial-time algorithm on trees (where both the degrees, and traversal costs are unbounded). We extend this algorithm to graphs where the difference between the number of edges and the number of vertices is constant, i.e. graphs that are in some sense close to trees. However, this extension does not cover cactus graphs, which have treewidth 2. To solve the problem for other special graph classes such as

cactus graphs or bounded treewidth graphs is one possible research direction. Another interesting research direction is to analyze these problems from the perspective of parameterized complexity where a few important and practical parameters are the treewidth of the graph, the number of colors, and the ratio of the largest traversal cost to the smallest non-zero traversal cost.

2 Preliminaries

Graphs, trees: Given an undirected graph $G = (V(G), E(G))$ and a vertex $v \in V(G)$, $\delta_G(v)$ denotes the set of edges incident to v in G , and $d_G(v) \stackrel{\text{def}}{=} |\delta_G(v)|$ is the degree of v in G . We denote a pair of vertices $u, v \in V(G)$ as uv , i.e. $uv \in E(G)$ if u and v are adjacent in G . The minimum and maximum degrees of G are defined as $\delta(G) \stackrel{\text{def}}{=} \min \{d_G(v) | v \in V(G)\}$ and $\Delta(G) \stackrel{\text{def}}{=} \max \{d_G(v) : v \in V(G)\}$. Given a tree T and two vertices $v_1, v_2 \in V(T)$, we denote by $P_T(v_1, v_2)$ the path between v_1 and v_2 in T . We denote a bipartite graph as a triple (V_1, V_2, E) where $\{V_1, V_2\}$ is the bipartition of its vertices and E is its edge set. Given two graphs G and G' , their union is $G \cup G' \stackrel{\text{def}}{=} (V(G) \cup V(G'), E(G) \cup E(G'))$.

The numbers of the inbound and outbound arcs of a vertex in a digraph are called its in-degree and out-degree, respectively. We denote the ordered pair (u, v) of vertices of G as uv , i.e. $uv \in E(G)$ if there is an arc from u to v in G . A digraph is a *rooted tree* or *arborescence* if its underlying graph is a tree and it contains a unique *root* vertex denoted by $\text{root}(T)$ which has a directed path to every other vertex of T . Each vertex $v \neq \text{root}(T)$ has in-degree 1. In this case we denote by $\text{in}_T(v)$ the unique inbound arc of v in T , and by $\text{parent}_T(v)$ the other endpoint of $\text{in}_T(v)$. The set $\text{chld}_T(v)$ is the set of all vertices u of T such that $\text{parent}_T(u) = v$. We also denote by $P_T(u, v)$ the unique path between vertices u and v in the tree T .

Reload and changeover costs: We consider proper edge colorings $\chi : E(G) \rightarrow X$ of a given graph G where the colors are taken from a set X and edges incident to the same vertex are assigned different colors. Without loss of generality we assume $X = \{1, 2, \dots, |X|\}$. Since, by Vizing's theorem, every graph is $\Delta(G) + 1$ edge colorable, we assume that $|X| \geq \Delta(G) + 1$ so that G is edge-colorable with colors from X .

The traversal costs are given by a nonnegative function $tc : X^2 \rightarrow \mathbb{R}^+ \cup \{0\}$ satisfying

- i) $tc(i, j) = tc(j, i)$ for every $i, j \in X$.

ii) $tc(i, i) = 0$ for every $i \in X$.

We denote as tc_{max} the maximum ratio between two positive traversal costs, i.e., $tc_{max} \stackrel{def}{=} \frac{\max\{tc_{i,j} | i, j \in X\}}{\min\{tc_{i,j} | i, j \in X, tc_{i,j} > 0\}}$. Let $P = (e_1, e_2, \dots, e_\ell)$ be a path of length ℓ of G . We denote by $tr(P) = \{(e_i, e_{i+1}) : 1 \leq i < \ell\}$ the set of traversals of P . The traversal cost associated with a traversal $t_i = (e_i, e_{i+1})$ of P with coloring χ is $tc_\chi(t_i) \stackrel{def}{=} tc(\chi(e_i), \chi(e_{i+1}))$. The traversal cost associated with P is $tc_\chi(P) \stackrel{def}{=} \sum_{t \in tr(P)} tc_\chi(t)$. Note that $tc_\chi(P) = 0$ whenever the length of P is zero or one, since in these cases $tr(P) = \emptyset$. Therefore, we assume that all paths under consideration have length at least 2.

Let \mathcal{P} be a set of paths. The set of traversals of \mathcal{P} is $tr(\mathcal{P}) \stackrel{def}{=} \bigcup_{P \in \mathcal{P}} tr(P)$. The *reload cost* of a set of paths \mathcal{P} with coloring χ is

$$rc_\chi(\mathcal{P}) \stackrel{def}{=} \sum_{P \in \mathcal{P}} tc_\chi(P) = \sum_{P \in \mathcal{P}} \sum_{t \in tr(P)} tc_\chi(t),$$

and its *changeover cost* is

$$cc_\chi(\mathcal{P}) \stackrel{def}{=} \sum_{t \in tr(\mathcal{P})} tc_\chi(t).$$

Note that the difference between $rc_\chi(\mathcal{P})$ and $cc_\chi(\mathcal{P})$ is that if a traversal occurs more than once, it contributes to $cc_\chi(\mathcal{P})$ only once, whereas every occurrence contributes to $rc_\chi(\mathcal{P})$.

Problem Statement: We assume without loss of generality that $E(G) = \bigcup_{P \in \mathcal{P}} E(P)$, i.e. every edge of G is used by at least one path. We note that whenever every traversal is in at most one path of \mathcal{P} , we have $rc_\chi(\mathcal{P}) = cc_\chi(\mathcal{P})$. Observe that, in particular, this holds when \mathcal{P} is a set of *distinct* paths with length 2. This simple fact will be useful throughout this work.

The minimum reload (resp. changeover) cost edge coloring (MINRCEC resp. MINCCEC) problem aims to find a proper edge coloring of G leading to a minimum reload (resp. changeover) cost with respect to \mathcal{P} . Formally,

MINRCEC/MINCCEC (\mathcal{P}, X, tc)

Input: A set of paths \mathcal{P} comprising a graph $G = \bigcup \mathcal{P}$, a set X of at least $\Delta(G) + 1$ colors, a traversal cost function $tc : X^2 \rightarrow \mathbb{R}^+ \cup \{0\}$

Output: A proper edge coloring $\chi : E(G) \rightarrow X$

Objective: Minimize $rc_\chi(\mathcal{P})/cc_\chi(\mathcal{P})$.

Given a tree T and a vertex $r \in V(T)$, let $\mathcal{P}(T, r) \stackrel{\text{def}}{=} \{P_T(r, v) : v \in V(T) \setminus \{r\}\}$ be the set of all paths between the root vertex r and all other vertices. The reload and changeover costs of T rooted at r are $rc_\chi(T, r) \stackrel{\text{def}}{=} rc_\chi(\mathcal{P}(T, r))$ and $cc_\chi(T, r) \stackrel{\text{def}}{=} cc_\chi(\mathcal{P}(T, r))$, respectively. Given a graph G and a vertex r of G , the minimum reload cost path tree edge coloring (MINRCPTEC) and minimum changeover cost arborescence edge coloring (MINCCAEC) problems aim to find a proper edge coloring of G and a spanning tree T rooted at r with minimum reload and changeover cost, respectively. Formally,

<p>MINRCPTEC/MINCCAEC (G, r, X, tc)</p> <p>Input: A graph G, a vertex r of G, a set X of at least $\Delta(G) + 1$ colors, a traversal cost function $tc : X^2 \rightarrow \mathbb{R}^+ \cup \{0\}$</p> <p>Output: A proper edge coloring $\chi : E(G) \rightarrow X$ and a spanning tree T of G rooted at r</p> <p>Objective: Minimize $rc_\chi(T, r)/cc_\chi(T, r)$.</p>
--

Approximation Algorithms, Reductions: Let Π be a minimization problem and $\rho \geq 1$. A (feasible) solution S of an instance I of Π is a ρ -*approximation* if the objective function value of S is at most ρ times the optimum. A polynomial-time algorithm ALG is a ρ -*approximation algorithm* for Π if ALG returns a ρ -approximation $ALG(I)$ for every instance I of Π . A *polynomial time approximation scheme* (PTAS) for Π is an infinite family of algorithms $\{ALG_\epsilon | \epsilon > 0\}$ such that ALG_ϵ is a $(1 + \epsilon)$ -approximation algorithm with running time $O(|I|^{h(\epsilon)})$ for some function h . PTAS also denotes the class of problems that admit a PTAS. APX is the class of problems that admit a c -approximation for some constant c .

Given two optimization problems Π and Π' with objective functions c_Π and $c_{\Pi'}$ respectively, an L -*reduction* from Π to Π' consists of two polynomial-time computable functions f, g such that a) f transforms every instance I of Π to an instance $f(I)$ of Π' , b) g transforms every solution s' of $f(I)$ to a solution $g(s')$ of I , c) $|OPT_{\Pi'}(f(I))| \leq \rho \cdot |OPT_\Pi(I)|$, and $|OPT_\Pi(I) - c_\Pi(g(s'))| \leq \rho' \cdot |OPT_{\Pi'}(f(I)) - c_{\Pi'}(s')|$ for two constants ρ, ρ' . A problem is in APX-Hard if every problem in APX can be reduced to it by an L -reduction. If a problem is in APX-Hard then it does not admit a PTAS unless $P = NP$. In this work, we use a different type of reduction that we will term LT -*reduction* or a *Turing type L -reduction*. An LT -reduction from Π to Π' is a polynomial-time com-

putable sequence of pairs of functions such that at least one of them is an L -reduction from Π to Π' . Clearly, by returning the best solution implied by the individual reductions, one can get a constant approximation to Π .

Biconnected Components and Block Trees: A *cut vertex* (*articulation point* or *separation vertex*) of a connected graph is a vertex whose removal (along with its incident edges) disconnects the graph. A graph with no articulation points is *biconnected*. A maximal biconnected induced subgraph of a graph is called a *biconnected component* or a *block* [11]. Any connected graph G can be decomposed in linear time into a tree whose vertices are the biconnected components of G and its articulation points. This tree is termed the *block tree* (or *superstructure*) of G . The edges of the block tree join every cut vertex to the blocks it belongs to [12] and every block to the cut vertices (of G) contained in it.

Matchings: A matching of a graph G is a subset $M \subseteq E(G)$ of pairwise nonadjacent edges. A matching is perfect if $V(M) = V(G)$. In an edge-weighted graph, minimum weight perfect matching is a perfect matching with minimum total edge weight and it can be computed in polynomial time.

The k -Lightest Subgraph Problem: The k -LIGHTEST SUBGRAPH problem is to find an induced subgraph H on k vertices, of a given edge-weighted graph G , with minimum total edge weight. This problem is NP-Hard in the strong sense even when the graph is a complete graph and the edge weights are either 1 or 2 [13].

The Minimum Set Cover Problem (MinSC): An instance of this problem is a set system $\mathcal{S} = \{S_1, S_2, \dots, S_m\}$, with $U \stackrel{\text{def}}{=} \cup \mathcal{S}$. Given such an instance, one has to find a subset $\mathcal{C} \subseteq \mathcal{S}$ that covers U , i.e. $\cup \mathcal{C} = U$, such that $|\mathcal{C}|$ is minimum. The special case in which $\forall i, |S_i| \leq k$, and $\forall u \in U, |\{S_i \in \mathcal{S} : u \in S_i\}| \leq \ell$, for two constants $k, \ell > 0$ is called the minimum (k, ℓ) -set cover problem. The minimum $(3, 2)$ -set cover problem (MIN3SC2) is APX-Hard [14], i.e. it does not admit a PTAS unless $P = NP$.

3 Hardness Results

In this section we show that MINCCEC and MINRCEC are inapproximable when the ratio κ_{tc} of the biggest traversal cost to the smallest non-zero traversal cost is unbounded. Then, we show that both problems remain NP-Hard in the strong sense even when $\kappa_{tc} = 2$ and G is a star.

We then return to the MINCCAEC and MINRCPTEC problems and show that they are APX-Hard in directed graphs even when $\kappa_{tc} = 2$.

Theorem 1. *MINCCEC and MINRCEC are inapproximable within any polynomial-time computable function $f(|\mathcal{P}|)$.*

Proof. The proof is by reduction from the chromatic index problem. The chromatic index of a graph G is either $\Delta(G)$ or $\Delta(G) + 1$. However, it is NP-Complete to decide between these two values [15]. Given a graph G we construct an instance $I = (\mathcal{P}, X, tc)$ where \mathcal{P} consists of all distinct paths of length 2 of G , $|X| = \Delta(G) + 1$, and

$$tc(i, j) = \begin{cases} 0 & \text{if } i = j \\ 1 & \text{if } i \neq j \text{ and } i, j \leq \Delta(G) \\ M & \text{otherwise} \end{cases}$$

where $M = |\mathcal{P}| \cdot f(|\mathcal{P}|)$. We recall that since the paths of \mathcal{P} are of length 2, we have $rc_\chi(\mathcal{P}) = cc_\chi(\mathcal{P})$ for every coloring χ . Assume, by way of contradiction, that there exists an $f(|\mathcal{P}|)$ -approximation algorithm \mathcal{A} for one of the problems. Then \mathcal{A} is an $f(|\mathcal{P}|)$ -approximation algorithm for both problems. If the chromatic index of G is $\Delta(G)$, let χ be a proper edge coloring of G using the first $\Delta(G)$ colors. Then all traversal costs are 1, and $rc_\chi(\mathcal{P}) = cc_\chi(\mathcal{P}) = |\mathcal{P}|$; therefore, the solution has value at most $|\mathcal{P}| \cdot f(|\mathcal{P}|)$. On the other hand, if the chromatic index of G is $\Delta(G) + 1$, then any edge coloring χ' uses $\Delta(G) + 1$ colors, and we have $rc_{\chi'}(\mathcal{P}) = cc_{\chi'}(\mathcal{P}) \geq |\mathcal{P}| + M - 1 = |\mathcal{P}| + |\mathcal{P}| \cdot f(|\mathcal{P}|) - 1 > |\mathcal{P}| \cdot f(|\mathcal{P}|)$ since there is at least one traversal with cost M . Therefore, G is $\Delta(G)$ edge-colorable if and only if \mathcal{A} returns a solution with cost at most $|\mathcal{P}| \cdot f(|\mathcal{P}|)$. \square

We now show that both problems are NP-Hard in the strong sense even in very simple graphs that are in particular $\Delta(G)$ -edge-colorable, namely stars, and have $\kappa_{tc} = 2$.

Theorem 2. *MINCCEC and MINRCEC are NP-Hard in the strong sense even when $tc(i, j) \in \{0, 1, 2\}$ for every pair $i, j \in X$ and G is a star.*

Proof. The proof is by reduction from the K-LIGHTEST SUBGRAPH problem, which is NP-Hard in the strong sense even on complete graphs with edge weights either 1 or 2 [13]. Given such an instance (K, w) of K-LIGHTEST SUBGRAPH where K is a complete graph on more than k vertices and w is the edge weight function such that w_{ij} is the weight of

the edge between vertices i and j , we build the following instance: G is a star on $k + 1$ vertices (k leaves), \mathcal{P} consists of the $\binom{k}{2}$ paths between every pair of leaves of G , $|X| = |K|$, and $tc(i, j) = w_{ij}$. Since all paths have length 2, we have $rc_\chi(\mathcal{P}) = cc_\chi(\mathcal{P})$ for every coloring χ . Moreover, $rc_\chi(\mathcal{P})$ is equal to the total edge weight of a clique on k vertices of K (corresponding to the set of k colors of X used in χ). \square

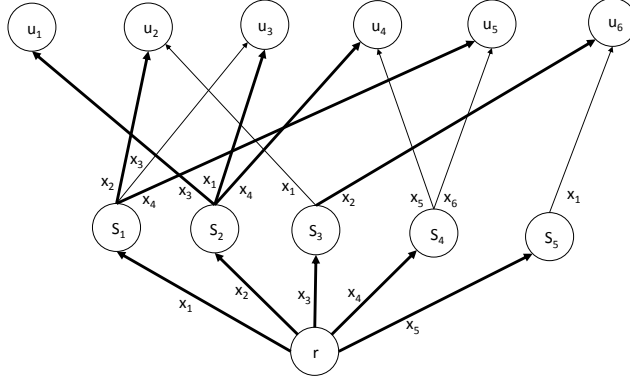


Fig. 1. The directed acyclic graph G corresponding to an instance of MINCCAEC with $S_1 = \{u_2, u_3, u_5\}$, $S_2 = \{u_1, u_3, u_4\}$, $S_3 = \{u_2, u_6\}$, $S_4 = \{u_4, u_5\}$, $S_5 = \{u_6\}$, $X_c = \{x_1, x_2, x_3, x_4\}$, and $X_e = \{x_5, x_6\}$. Bold arcs indicate the spanning tree corresponding to a minimum set cover $\mathcal{C}^* = \{S_1, S_2, S_3\}$.

Theorem 3. MINCCAEC and MINRCPTEC are APX-Hard in directed graphs even when $tc(i, j) \in \{0, 1, 2\}$ for every pair $i, j \in X$.

Proof. The proof is by *LT*-reduction from MIN3SC2, which is known to be APX-Hard. We consider an instance \mathcal{S} of MIN3SC2 with n elements, $m \geq 4$ sets and with every set cover consisting of at least 4 sets. Otherwise, an optimal set cover can be found in polynomial time. Given such an instance and an integer $k \leq m$, we construct an instance $f_k(\mathcal{S}) = (G, r, X, tc)$ as follows (see Figure 1). $G = (V, E)$ is a directed

acyclic graph with $V = \{r\} \cup \mathcal{S} \cup U$ where $U = \cup \mathcal{S}$, $E = E_1 \cup E_2$, $E_1 = \{rS_i \mid S_i \in \mathcal{S}\}$ and $E_2 = \{S_i u_j \mid S_i \in \mathcal{S}, u_j \in S_i\}$. The color set is the disjoint union X of two color sets X_c and X_e with $|X_c| = k$ and $|X_e| = \Delta(G) + 1 - k$. Finally,

$$tc(x, y) = \begin{cases} 0 & \text{if } x = y \\ 1 & \text{if } x \neq y \text{ and } x, y \in X_c \\ 2 & \text{otherwise.} \end{cases}$$

We observe that $rc_\chi(T, r) = cc_\chi(T, r)$ since every directed path has length at most 2. Therefore, our reduction behaves in the same way for MINCCAEC and MINRCPTEC. For any feasible solution (T, χ) of $f_k(\mathcal{S})$, the set of parents of the vertices U in T is a set cover $g_k(T, \chi)$. This completes the description of the reduction.

Let \mathcal{C}^* be a minimum set cover of \mathcal{S} and $k^* = |\mathcal{C}^*|$. We now show that f_{k^*}, g_{k^*} is an L -reduction. Consider the subgraph of G induced by the vertices $\{r\} \cup \mathcal{C}^* \cup \{r\}$. All the vertices of this graph have degree at most 4, except r whose degree is $k^* \geq 4$. Then this subgraph is bipartite with maximum degree k^* . Therefore, we can color all the arcs of this subgraph with the k^* colors of X_c , and the remaining arcs (all incident to r) with colors from X_e . The cost of this solution is n , thus $OPT(f_{k^*}(\mathcal{S})) \leq n$. Finally, a spanning tree T is built by joining every vertex u_i to an arbitrary vertex $S_j \in \mathcal{C}^*$ such that $u_i \in S_j$ and each u_i is a leaf of the spanning tree. We conclude that

$$OPT(f_{k^*}(\mathcal{S})) \leq n \leq 3 \cdot OPT(\mathcal{S}) \quad (1)$$

where the last inequality holds since every set can cover at most three elements.

Let (T, χ) be a solution of $f_{k^*}(\mathcal{S})$. We partition $g_{k^*}(T, \chi)$ into two sets $\mathcal{S}_c = \{S_i \in g_{k^*}(T, \chi) \mid \chi(rS_i) \in X_c\}$ and $\mathcal{S}_e = \{S_i \in g_{k^*}(T, \chi) \mid \chi(rS_i) \in X_e\}$. We observe that $|\mathcal{S}_c| \leq k^*$ since χ colors the inbound arcs of \mathcal{S}_c with distinct colors of X_c (all these arcs are incident to r) and $|X_c| = k^*$. Therefore,

$$|g_{k^*}(T, \chi)| - OPT(\mathcal{S}) = |g_{k^*}(T, \chi)| - k^* = |\mathcal{S}_e| + |\mathcal{S}_c| - k^* \leq |\mathcal{S}_e|.$$

We have that $cc_\chi(T, r) \geq n + |\mathcal{S}_e|$ since the arc leading to u_i in T incurs a traversal cost of at least 1 in the parent S_j of u_i and for every $S_j \in \mathcal{S}_e$ there is at least one traversal that costs 2. Therefore,

$$|\mathcal{S}_e| \leq cc_\chi(T, r) - n \leq cc_\chi(T, r) - OPT(f_{k^*}(\mathcal{S})).$$

We combine the last two inequalities to get

$$|g_{k^*}(T, \chi)| - OPT(\mathcal{S}) \leq cc_\chi(T, r) - OPT(f_{k^*}(\mathcal{S})). \quad (2)$$

By inequalities (1) and (2), f_{k^*} and g_{k^*} constitute an L -reduction, as required. \square

4 Polynomial-time Solvable Cases

In this section we present polynomial-time algorithms for some special cases. We start with the definitions and notations used in this section. We denote the set of all proper edge colorings of a graph G by \mathcal{F}_G . Two partial functions f, f' agree if $f(x) = f'(x)$ whenever both f and f' are defined on x . We denote this fact by $f \sim f'$.

The following discussion refers to the changeover cost; however, it holds for the reload cost, too. Whenever this is not the case, the difference between the two costs will be made explicit. For a subgraph H of G we say that a traversal (e_i, e_j) is *within* H if $e_i, e_j \in E(H)$, and we denote by $tr(\mathcal{P}, H)$ the set of traversals of \mathcal{P} within H . We define $rc_\chi(\mathcal{P}, H)$ and $cc_\chi(\mathcal{P}, H)$ similarly by taking into account only traversals within H . Let H_1, H_2, \dots, H_k be subgraphs of G such that every traversal is within exactly one subgraph H_i . Clearly, $cc_\chi(\mathcal{P}) = \sum_{i=1}^k cc_\chi(\mathcal{P}, H_k)$.

In the sequel, we analyze the decomposition of a spanning tree T of G (and its cost) by the block tree of G . Let T be a spanning tree of G rooted at some vertex r . For the MINRCPTEC and MINCCAEC problems, it is convenient to choose the root as the vertex r given in the instance. Consult Figure 2 for the following discussion. For a non-root vertex v of T , we denote by T_v the subtree of T rooted at v with the addition of the arc $e = in_T(v)$. Let S_v denote the star consisting of v and its incident edges. Moreover, let $chld_T(v) = \{v_1, \dots, v_k\}$ and $e_i = vv_i$. Clearly, every traversal within T_v is either within S_v or within T_{v_i} for some $i \in [k]$. Therefore,

$$cc_\chi(\mathcal{P}, T_v) = cc_\chi(\mathcal{P}, S_v) + \sum_{i=1}^k cc_\chi(\mathcal{P}, T_{v_i}). \quad (3)$$

We denote by $OPT_{cc}(\mathcal{P}, v, x)$ the minimum changeover cost within T_v , among all colorings χ such that $\chi(in_T(v)) = x$. Formally,

$$OPT_{cc}(\mathcal{P}, v, x) \stackrel{def}{=} \min \{cc_\chi(\mathcal{P}, T_v) : \chi \in \mathcal{F}_{T_v}, \chi(in_T(v)) = x\}. \quad (4)$$

Since T_v does not contain any traversals whenever v is a leaf, we have

$$OPT_{cc}(\mathcal{P}, v, x) = 0 \text{ whenever } v \text{ is a leaf.} \quad (5)$$

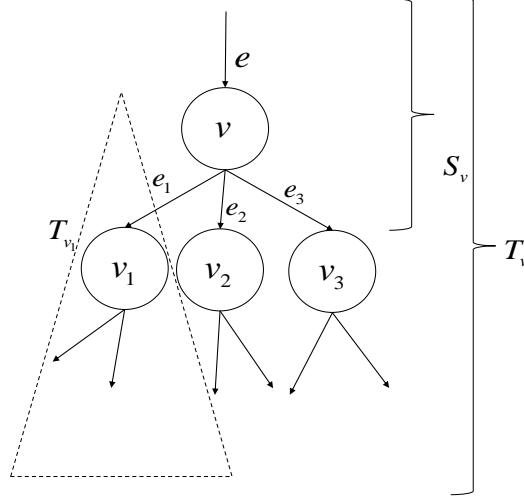


Fig. 2. Notation for Section 4.

In order to compute $OPT_{cc}(\mathcal{P}, v, x)$ we categorize all proper edge colorings χ of T_v by the colorings χ_v they induce on S_v :

$$\begin{aligned} \alpha_{cc}(\chi_v) &\stackrel{def}{=} \min \{cc_\chi(\mathcal{P}, T_v) : \chi \in \mathcal{F}_{T_v}, \chi \sim \chi_v\}, \\ OPT_{cc}(\mathcal{P}, v, x) &= \min \{\alpha_{cc}(\chi_v) : \chi_v \in \mathcal{F}_{S_v}, \chi_v(e) = x\}. \end{aligned} \quad (6)$$

where $\alpha_{cc}(\chi_v)$ is the minimum cost within T_v among all colorings that induce the coloring χ_v on S_v . We define α_{rc} similarly. For a fixed coloring

$\chi_v \in \mathcal{F}_{S_v}$, we proceed as follows by first using (3):

$$\begin{aligned}
\alpha_{cc}(\chi_v) &= \min \left\{ cc_{\chi}(\mathcal{P}, S_v) + \sum_{i=1}^k cc_{\chi}(\mathcal{P}, T_{v_i}) : \chi \in \mathcal{F}_{T_v}, \chi \sim \chi_v \right\} \\
&= cc_{\chi_v}(\mathcal{P}, S_v) + \sum_{i=1}^k \min \{ cc_{\chi}(\mathcal{P}, T_{v_i}) : \chi \in \mathcal{F}_{T_v}, \chi \sim \chi_v \} \\
&= cc_{\chi_v}(\mathcal{P}, S_v) + \sum_{i=1}^k \min \{ cc_{\chi}(\mathcal{P}, T_{v_i}) : \chi \in \mathcal{F}_{T_v}, \chi(e_i) = \chi_v(e_i) \} \\
&= cc_{\chi_v}(\mathcal{P}, S_v) + \sum_{i=1}^k OPT_{cc}(\mathcal{P}, v_i, \chi_v(e_i)) \\
&= cc_{\chi_v}(\mathcal{P}, S_v) + \sum_{v' \in \text{chld}_T(v)} OPT_{cc}(\mathcal{P}, v', \chi_v(vv')), \tag{7}
\end{aligned}$$

where the equality second to last holds by (4). In particular, for $v = r$ we obtain the optimum of the instance as:

$$cc^*(\mathcal{P}) = \min \{ \alpha_{cc}(\chi_r) : \chi_r \in \mathcal{F}_{S_r} \}. \tag{8}$$

Equations (5) through (8) imply Algorithm 1, which is a dynamic programming algorithm for the case when G is a tree. The number of entries $OPT_{cc}(\mathcal{P}, v, x)$ computed by the algorithm is $|V(G)| \cdot |X|$, i.e. polynomial in the size of the input. In order to get a polynomial-time algorithm, we have to compute every entry in polynomial time. The value $cc_{\chi_v}(\mathcal{P}, S_v)$ can be computed in time $O(|\mathcal{P}|)$ since every path has at most one traversal in S_v . Therefore, $\alpha_{cc}(\chi_v)$ can be computed in time $O(|\mathcal{P}| + k) = O(|\mathcal{P}| + \Delta(G))$. In the sequel, we analyze the computation time of $OPT_{cc}(\mathcal{P}, v, x)$ in various cases.

Algorithm 1 Dynamic Programming for MINCCEC in Trees

Require: Input (\mathcal{P}, X, tc)
Require: $\cup \mathcal{P}$ is a tree
 $T \leftarrow \cup \mathcal{P}$
 Root T at an arbitrary vertex r
for all vertex $v \in V(T) \setminus \{r\}$ in a postorder traversal of T **do**
 for all $x \in X$ **do**
 if v is a leaf of T **then**
 $OPT_{cc}(\mathcal{P}, v, x) \leftarrow 0$
 else
 $OPT_{cc}(\mathcal{P}, v, x) \leftarrow \text{COMPUTENONLEAF}(\mathcal{P}, v, x)$
return $cc^*(\mathcal{P})$ using Equation (8)

function $\text{COMPUTENONLEAF}(\mathcal{P}, v, x)$
 $e \leftarrow in_T(v)$
 $min \leftarrow \infty$
 for $\chi_v \in \mathcal{F}_{S_v} \wedge \chi_v(e) = x$ **do**
 $cc_{\chi_v}(\mathcal{P}, S_v) \leftarrow \sum_{t \in tr(\mathcal{P}, S_v)} tc_{\chi_v}(t)$
 Compute $\alpha_{cc}(\chi_v)$ using Equation (7)
 if $\alpha_{cc}(\chi_v) < min$ **then**
 $min \leftarrow \alpha_{cc}(\chi_v)$
return min

Theorem 4. MINCCEC and MINRCEC are solvable in polynomial time when G is a tree and $|X|^{\Delta(G)}$ is polynomial in the input size.

Proof. The number of proper edge colorings χ_v of S_v using colors from X such that $\chi_v(e) = x$ is at most $|X|^{\Delta(G)}$. Therefore, the computation time of $OPT_{cc}(\mathcal{P}, v, x)$ (in Function COMPUTENONLEAF) is at most $O((|\mathcal{P}| + \Delta(G))|X|^{\Delta(G)})$, i.e. polynomial in the input size. \square

Corollary 1. MINCCEC and MINRCEC are solvable in polynomial time whenever G is a bounded degree tree.

Note that Corollary 1 complements Theorem 2, which proves that MINCCEC and MINRCEC are NP-Hard for unbounded degree stars. In the sequel, we show that MINCCEC and MINRCEC are polynomial-time solvable in trees and graphs having a structure close to a tree, i.e. graphs G where $|E(G)| - |V(G)|$ is bounded by some constant.

Theorem 5. MINCCEC and MINRCEC are solvable in polynomial time when G is a tree and a particular vertex r is an endpoint of every path $P \in \mathcal{P}$.

Proof. We consider G as rooted at r . We observe that since all paths have an endpoint at r , all traversals within S_v contain the edge $e = in_T(v)$. Therefore, for $\chi_v(e) = x$,

$$cc_{\chi_v}(\mathcal{P}, S_v) = \sum_{i=1}^k tc(\chi_v(e), \chi_v(e_i)) = \sum_{i=1}^k tc(x, \chi_v(e_i))$$

and

$$rc_{\chi_v}(\mathcal{P}, S_v) = \sum_{i=1}^k tc(x, \chi_v(e_i)) |\mathcal{P}_{e_i}|$$

where \mathcal{P}_{e_i} is the set of paths of \mathcal{P} that contain e_i . Substituting in (7) we get

$$\alpha_{cc}(\chi_v) = \sum_{i=1}^k (tc(x, \chi_v(e_i)) + OPT_{cc}(\mathcal{P}, v_i, \chi_v(e_i))),$$

and similarly

$$\alpha_{rc}(\chi_v) = \sum_{i=1}^k (tc(x, \chi_v(e_i)) |\mathcal{P}_{e_i}| + OPT_{rc}(\mathcal{P}, v_i, \chi_v(e_i))).$$

We now observe that these values can be computed in polynomial time by Function COMPUTE NONLEAF in Algorithm 2, as described in the sequel. Consider the complete bipartite graph B where the bipartition of the vertices is $\{\{v_1, \dots, v_k\}, X - x\}$. There is a one-to-one correspondence between the proper edge colorings χ_v of S_v with $\chi_v(e) = x$ and the matchings of B with size k . The matching M_{χ_v} corresponding to the edge coloring χ_v is such that $v_i y \in M_{\chi_v}$ if and only if $\chi_v(e_i) = y$. We assign the weight $w(v_i y) = tc(x, y) + OPT_{cc}(\mathcal{P}, v_i, y)$ to the edge $v_i y$ for every $i \in [k]$ and $y \in X - x$. Under this setting, the total weight of the matching M_{χ_v} corresponding to χ_v is equal to $\alpha_{cc}(\chi_v)$. We conclude that minimizing $\alpha_{cc}(\chi_v)$ is equivalent to finding a minimum weight matching with k edges on this weighted graph, i.e. $OPT_{cc}(\mathcal{P}, v, x)$ can be computed in polynomial time. \square

Algorithm 2 COMPUTE NONLEAF by Minimum Weight Perfect Matching

Require: $\cup \mathcal{P}$ is a tree

Require: All paths in \mathcal{P} have a common endpoint

function COMPUTE NONLEAF(\mathcal{P}, v, x)

$e \leftarrow \text{in}_T(v)$

 Construct a complete bipartite graph B with bipartition $\{\text{chld}_T(v), X \setminus \{x\}\}$

for $v' \in \text{chld}_T(v)$ **do**

for $y \in X - x$ **do**

$w(v'y) \leftarrow tc(x, y) + \text{OPT}_{cc}(\mathcal{P}, v', y)$

return The minimum weight of a perfect matching of B .

We note that in the special case of MINCCAEC (resp. MINRCPTEC) problem when G is a tree, there is only one spanning tree; therefore, we get a special case of the MINCCEC (resp. MINRCEC) problem when G is a tree and a particular vertex r of G is the source vertex of all paths. Therefore,

Corollary 2. MINCCAEC and MINRCPTEC are solvable in polynomial time for trees.

Consider an input graph G that is not a tree, but a tree can be obtained by the removal of at most c edges from G where c is a constant. Then, one can try all of the at most $|E|^c$ combinations of the edges to be removed, solve the problem for each remaining tree in turn, and return the best solution. Therefore,

Corollary 3. MINCCAEC and MINRCPTEC are solvable in polynomial time for graphs G where $|E(G)| - |V(G)|$ is bounded by some constant.

The following theorem extends this idea for graphs with sparse blocks and bounded degree cut vertices.

Theorem 6. MINCCAEC and MINRCPTEC are solvable in polynomial time whenever

- a) the degree of every cut vertex of G is bounded by some constant c_1 ,
and
- b) $|E(B)| - |V(B)|$ is bounded by some constant c_2 for every block B of G .

Proof. **Block Tree Notation:** Consult Figure 3 for the definitions and notation we introduce in this paragraph. For simplicity, we modify the

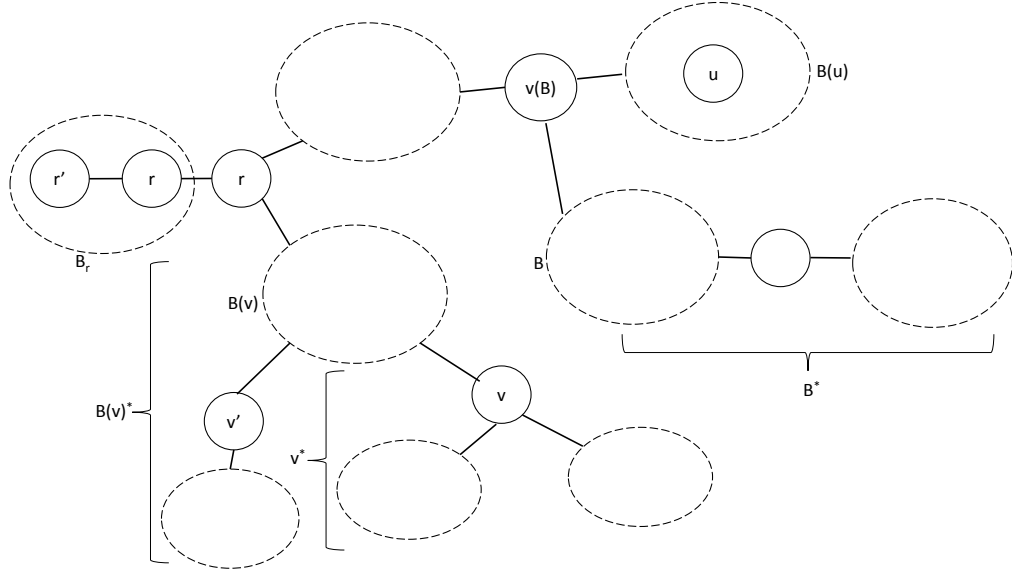


Fig. 3. Notation regarding the vertices of a block tree.

instance as follows. We add to G an additional vertex r' incident only to r , and an additional color $0 \notin X$ that is usable only in the new edge rr' , and $tc(0, i) = 0$ for every $i \in X$. Moreover, we set r' as the root of the modified instance. Clearly, every solution of the new instance contains rr' and there exists an optimal solution in which rr' is colored 0. After this modification, r is a cut vertex of G and $B_r = \{r, r'\}$ is a block of G . We consider the block tree \mathcal{T} of G as rooted at B_r . We recall that the neighbours of a cut vertex (resp. block) in \mathcal{T} are blocks (resp. cut vertices). For a cut vertex v (resp. block B) of G , we denote by $B(v)$ (resp. $v(B)$) the parent of v (resp. B) in \mathcal{T} . For a non-cut vertex u of G , we denote by $B(u)$ the unique block that contains u . For a block B (resp. cut vertex v), we denote by B^* (resp. v^*) the set of all vertices of G contained in the blocks of the subtree of \mathcal{T} rooted at B (resp. v). Note that $B(v) \cup v^* \subseteq B(v)^*$.

Spanning Trees and the Block Tree: For a set U of vertices of G , we denote by $\mathcal{S}(U)$ the set of subgraphs of G that are trees and span U , i.e. the set of trees (U, E_T) such that $E_T \subseteq E(G)$. Note that possibly $\mathcal{S}(U) = \emptyset$. In particular, $\mathcal{S}(G) \stackrel{\text{def}}{=} \mathcal{S}(V(G))$ is the set of spanning trees of G . Let T be a spanning tree of G rooted at r' . We note that graph

$T[B]$ induced by T on a block B of G is a spanning tree of B rooted at $v(B)$. For a spanning tree T and a cut vertex v of G , we denote by T_{v^*} the subtree of T that contains the vertices v^* and the parent of v , i.e. $T_{v^*} = T[v^* \cup \text{parent}_T(v)]$. Note that T_{v^*} is a subtree of T_v , but possibly different from T_v since v might have descendants in $B(v)$ and such vertices are not in v^* .

Structure of the Dynamic Programming Algorithm: We now present a dynamic programming algorithm for the problem (see pseudocode in Algorithm 3). We first introduce the values to be computed by the algorithm. For every cut vertex v of G and a color $x \in X$, the algorithm computes $OPT_{cc}(v, x)$ that denotes the minimum changeover cost within T_{v^*} of a spanning tree T of G and a coloring χ such that $\chi(\text{in}_T(v)) = x$, i.e.

$$OPT_{cc}(v, x) \stackrel{\text{def}}{=} \min \{cc_\chi(T_{v^*}) : T \in \mathcal{S}(G), \chi(\text{in}_T(v)) = x\}.$$

Clearly, the optimum of the instance is $OPT_{cc}(r, 0)$.

Given a block B and a spanning tree $\hat{T} \in \mathcal{S}(B)$ of it, we consider \hat{T} as rooted at $v(B)$. For every such block B , spanning tree \hat{T} , and every non-root vertex of \hat{T} (i.e. every vertex $v \in B \setminus \{v(B)\}$) the algorithm computes the value $OPT_{cc}(v, \hat{T}, x)$, which denotes the minimum changeover cost within T_v of a spanning tree T of G that induces the tree \hat{T} on $B(v)$ and a coloring χ such that $\chi(\text{in}_T(v)) = \chi(\text{in}_{\hat{T}}(v)) = x$, i.e.

$$OPT_{cc}(v, \hat{T}, x) \stackrel{\text{def}}{=} \min \left\{ cc_\chi(T_v) : T \in \mathcal{S}(G), T[B(v)] = \hat{T}, \chi(\text{in}_{\hat{T}}(v)) = x \right\}$$

for every $v \in V(B) \setminus \{v(B)\}$, $\hat{T} \in \mathcal{S}(B)$, and $x \in X \setminus \{0\}$. A spanning tree of B is obtained by removing $|E(B)| - |V(B)| + 1 \leq c_2 + 1$ edges from $E(B)$. Therefore, $|\mathcal{S}(B)| \leq \binom{|E(B)|}{c_2 + 1} \leq |E(B)|^{c_2 + 1}$. Therefore, the total number of values to be computed is $\mathcal{O}(|V(G)| \cdot |X| \cdot |E(G)|^{c_2 + 1})$, which is polynomial in the input size. It remains to show a) how to compute each value in time polynomial in the input size, and b) how to compute the optimum once these values are computed.

We perform a bottom-up traversal of \mathcal{T} during which we compute, at a block B , the values $OPT_{cc}(v, \hat{T}, x)$, and at a cut vertex v , the values $OPT_{cc}(v, x)$.

Algorithm for a block: We start with the description of the computation at a block B . Consult Figure 4 for this description. Let B be a block of G , $\hat{T} \in \mathcal{S}(B)$, and T a spanning tree of G such that $T[B] = \hat{T}$. We

perform a bottom-up traversal of \hat{T} . At each non-root vertex v of \hat{T} , (i.e. $v \in B - v(B)$) we proceed as follows. Let $e = in_T(v) = in_{\hat{T}}(v)$, $chld_T(v) = \{v_1, \dots, v_k\}$, and $e_i = vv_i$ for $i \in [k]$. We first assume, for simplicity, that v is not a cut vertex. In this case, $chld_T(v) \subseteq B - v(B)$, and we can compute $OPT_{cc}(v, \hat{T}, x)$, in the same way that we did in Theorem 5, namely by reducing the problem to finding a minimum weight perfect matching $\gamma(H, w)$ of the bipartite graph H with bipartition $\{X - x, \{e_1, \dots, e_k\}\}$ and weights $w(e_i y) = tc(x, y) + OPT_{cc}(v_i, \hat{T}, y)$. Now assume that v is a cut vertex, and let $\{v_1, \dots, v_{k'}\}$ be the set of its children in B where $k' < k$. In this case, we divide T_v into the subtrees $T_{v_1}, \dots, T_{v_{k'}}$ and T_{v^*} . We note that all these trees intersect exactly at v and that $cc_\chi(T_v)$ is the sum of the costs of these trees with the addition of the traversal costs from e to each e_i . Therefore, the optimum cost can be computed by adding $OPT_{cc}(v, x)$ to $\gamma(H, w)$. Summarizing,

$$OPT_{cc}(v, \hat{T}, x) = \gamma(H, w) + \begin{cases} OPT_{cc}(v, x) & \text{if } v \text{ is a cut vertex} \\ 0 & \text{otherwise,} \end{cases}$$

where $\gamma(H, w)$ is the minimum weight perfect matching of the complete bipartite graph H with bipartition $\{X - x, chld_{\hat{T}}(v)\}$ and edge weights

$$w(e_i y) = tc(x, y) + OPT_{cc}(v_i, \hat{T}, y).$$

At this point we note that for the MINRCPTec problem the weights of H are $w(e_i y) = tc(x, y) \cdot |\mathcal{P}_{e_i}| + OPT_{rc}(v_i, \hat{T}, y)$. This computation can be clearly carried out in polynomial time as in the case of Theorem 5.

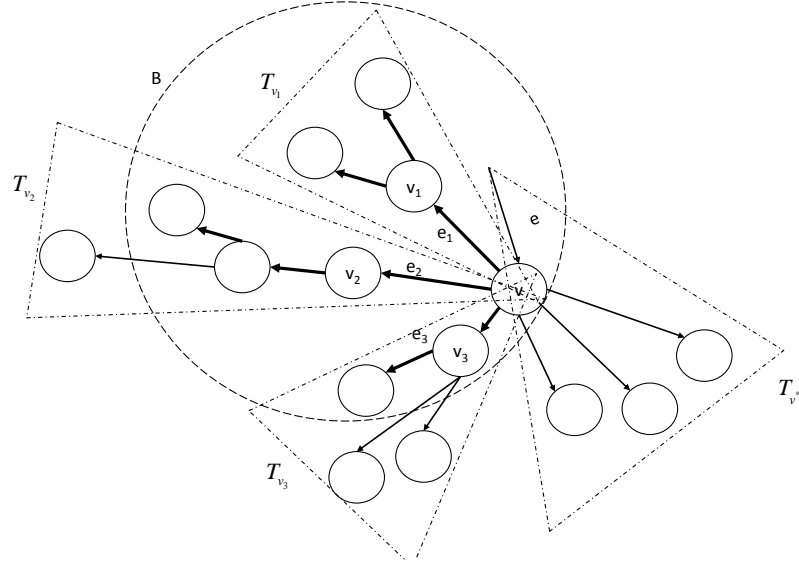


Fig. 4. The computation of the value $OPT_{cc}(v, \hat{T}, x)$. The bold arcs depict the arcs of \hat{T} .

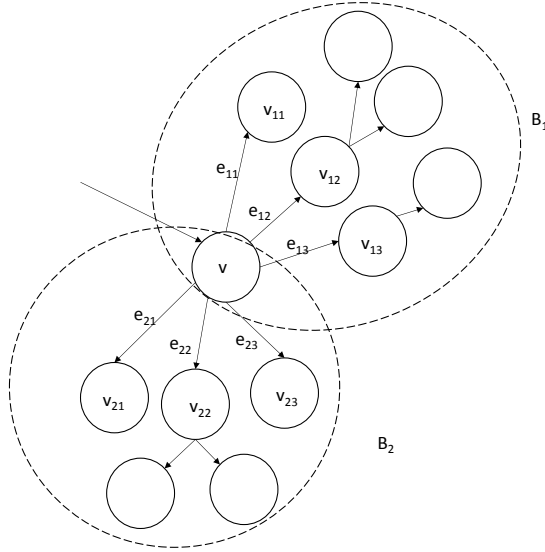


Fig. 5. The computation of the value $OPT_{cc}(v, x)$ for a cut vertex v .

Algorithm 3 Dynamic Programming for MINCCEC in Graphs with Sparse Blocks and Bounded Degree Cut Vertices

```

 $V(G) \leftarrow V(G) + r'$ .
 $E(G) \leftarrow E(G) + rr'$ .
 $B_r \leftarrow \{r, r'\}$ .
 $X \leftarrow X + 0$ .
for all  $x \in X$  do
     $tc(0, x) \leftarrow 0$ .
 $\mathcal{T} \leftarrow$  the block tree of  $G$  rooted at  $B_r$ 

for all vertices  $U \in V(\mathcal{T}) \setminus \{B_r\}$  in a postorder traversal of  $\mathcal{T}$  do
    if  $U$  is a block  $B$  of  $G$  then
        for all  $\hat{T} \in \mathcal{S}(B)$  rooted at  $v(B)$  do
            COMPUTEBLOCK( $B, \hat{T}$ )
    else  $\triangleright U$  is a cut vertex  $v$  of  $G$ 
        for all  $x \in X$  do
            Compute  $OPT_{cc}(v, x)$  using equation (9)
return  $OPT_{cc}(r, 0)$ .

function COMPUTEBLOCK( $B, \hat{T}$ )
    for all vertices  $v \in V(\hat{T}) \setminus \{v(B)\}$  in a postorder traversal of  $\hat{T}$  do
        for all  $x \in X$  do
            Construct a complete bipartite graph  $H$  with
                bipartition  $\{chld_{\hat{T}}(v), X - x\}$ 
            for  $v' \in chld_{\hat{T}}(v)$  do
                for  $y \in X - x$  do
                     $w(v'y) \leftarrow tc(x, y) + OPT_{cc}(v', \hat{T}, y)$ 
             $OPT_{cc}(v, \hat{T}, x) \leftarrow \gamma(H, w)$   $\triangleright$  The min. weight perfect matching
            if  $v$  is a cut vertex of  $G$  then
                 $OPT_{cc}(v, \hat{T}, x) \leftarrow OPT_{cc}(v, \hat{T}, x) + OPT_{cc}(v, x)$ .

```

Algorithm for a cut vertex: We proceed with the description of the computation of the values $OPT_{cc}(v, x)$ for a cut vertex v of G , with $chld_{\mathcal{T}}(v) = \{B_1, \dots, B_k\}$ (see Figure 5). We perform an exhaustive search by guessing a) the coloring χ_v that an optimal solution χ induces on the edges incident to v , and b) the spanning trees $\hat{T}_i = T[B_i]$ for every child block B_i of v . Note that v is the root of all the trees \hat{T}_i . Let $chld_{\hat{T}_i}(v) =$

$\{v_{i1}, \dots, v_{ik_i}\}$ and $e_{ij} = vv_{ij}$. Then, recalling that $\chi(\text{in}_T(v)) = x$, we have

$$\begin{aligned} cc_\chi(T_v) &= \sum_{i=1}^k \sum_{j=1}^{k_i} (tc(x, \chi_v(e_{ij})) + cc_\chi(T_{v_{ij}})) \\ &= \sum_{i=1}^k \sum_{j=1}^{k_i} tc(x, \chi_v(e_{ij})) + \sum_{i=1}^k \sum_{j=1}^{k_i} cc_\chi(T_{v_{ij}}). \end{aligned}$$

Given a guess for χ_v and \hat{T}_i , the first sum is fixed and the trees $T_{v_{ij}}$ are pairwise disjoint. Therefore, each term in the second summation can be minimized independently. Then, the minimum for a given guess is

$$\begin{aligned} &\sum_{i=1}^k \sum_{j=1}^{k_i} tc(x, \chi_v(e_{ij})) + \sum_{i=1}^k \sum_{j=1}^{k_i} \min cc_\chi(T_{v_{ij}}) \\ &= \sum_{i=1}^k \sum_{j=1}^{k_i} tc(x, \chi_v(e_{ij})) + \sum_{i=1}^k \sum_{j=1}^{k_i} OPT_{cc}(v_{ij}, \hat{T}_i, \chi_v(e_{ij})) \\ &= \sum_{i=1}^k \sum_{j=1}^{k_i} \left(tc(x, \chi_v(e_{ij})) + OPT_{cc}(v_{ij}, \hat{T}_i, \chi_v(e_{ij})) \right). \end{aligned}$$

For a given guess of χ_v , the minimum over all guesses of the subtrees \hat{T}_i is

$$\sum_{B \in \text{chld}_T(v)} \min_{\hat{T} \in \mathcal{S}(B)} \sum_{v' \in \text{chld}_{\hat{T}}(v)} \left(tc(x, \chi_v(vv')) + OPT_{cc}(v', \hat{T}, \chi_v(vv')) \right).$$

Minimizing over all guesses of χ_v we get

$$\begin{aligned} OPT_{cc}(v, x) &= \min_{\chi_v \in \mathcal{F}(S_v)} \sum_{B \in \text{chld}_T(v)} \min_{\hat{T} \in \mathcal{S}(B)} \sum_{v' \in \text{chld}_{\hat{T}}(v)} \\ &\quad \left(tc(x, \chi_v(vv')) + OPT_{cc}(v', \hat{T}, \chi_v(vv')) \right), \quad (9) \end{aligned}$$

where $\mathcal{F}(S_v)$ is the set of all colorings χ_v of the edges incident to v using colors from X with the exception that $\chi_r(rr') = 0 \notin X$. We note that for the MINRCPTEC problem the innermost term on the right hand side becomes $tc(x, \chi_v(vv')) \cdot |\mathcal{P}_{vv'}| + OPT_{rc}(v', \hat{T}, \chi_v(vv'))$.

The number of terms in the inner summation is at most the degree $d(v)$ of the cut vertex v , which is at most c_1 . The number of guesses is at most $|\mathcal{F}(S_v)| \cdot |\mathcal{S}(B_i)| \leq |X|^{d(v)} \cdot |E(G)|^{c_2+1} \leq |X|^{c_1} \cdot |E(G)|^{c_2+1}$. Therefore, $OPT_{cc}(v, x)$ can be computed in time polynomial in the input size. \square

References

1. H.C. Wirth and J. Steffan. Reload cost problems: minimum diameter spanning tree. *Discrete Applied Mathematics*, 113(1):73–85, 2001.
2. G. Galbiati. The complexity of a minimum reload cost diameter problem. *Discrete Applied Mathematics*, 156(18):3494–3497, 2008.
3. Giulia Galbiati, Stefano Gualandi, and Francesco Maffioli. On minimum reload cost cycle cover. *Discrete Applied Mathematics*, 164(1):112–120, 2014.
4. Laurent Gourvès, Adria Lyra, Carlos Martinhon, and Jérôme Monnot. The minimum reload s-t path, trail and walk problems. *Discrete Applied Mathematics*, 158(13):1404–1417, July 2010.
5. E. Amaldi, Giulia Galbiati, and Francesco Maffioli. On minimum reload cost paths, tours, and flows. *Networks*, 57(3):254–260, 2011.
6. I. Gamvros, L. Gouveia, and S. Raghavan. Reload cost trees and network design. *Networks*, 59(4):365–379, 2012.
7. Giulia Galbiati, Stefano Gualandi, and Francesco Maffioli. On minimum changeover cost arborescences. In *Experimental Algorithms - 10th International Symposium (SEA), Kolimpari, Chania, Crete, Greece*, pages 112–123, May 2011.
8. Didem Gözüpek, Mordechai Shalom, Ariella Voloshin, and Shmuel Zaks. On the complexity of constructing minimum changeover cost arborescences. *Theoretical Computer Science*, 540:40–52, June 2014.
9. Didem Gozupek, Seyed Buhari, and Fatih Alagoz. A spectrum switching delay-aware scheduling algorithm for centralized cognitive radio networks. *IEEE Transactions on Mobile Computing*, 12(7):1270–1280, 2013.
10. Didem Gözüpek, Hadas Shachnai, Mordechai Shalom, and Shmuel Zaks. Constructing minimum changeover cost arborescences in bounded treewidth graphs. *Theoretical Computer Science*, 2016. to appear.
11. J.A. Bondy and U.S.R. Murty. *Graph theory*. Springer, 2008.
12. Shimon Even. *Graph Algorithms*. Computer Science Press, 1979.
13. Rémi Watrigant, Marin Bougeret, and Rodolphe Giroudeau. The k-sparsest subgraph problem. Technical report, RR-12019, 2012.
14. R. Duh and M. Fürer. Approximation of k-set cover by semi-local optimization. In *ACM Symposium on Theory of Computing (STOC)*, pages 256–264, 1997.
15. Ian Holyer. The np-completeness of edge-coloring. *SIAM Journal on Computing*, 10(4):718–720, 1981.